International Journal of Pure and Applied Mathematics

Volume 18 No. 3 2005, 271-283

MODELING SCHEDULED DATAFLOW ARCHITECTURE: AN OPEN QUEUING NETWORK MODEL APPROACH

Vidhyacharan Bhaskar[1], Laurie L. Joiner[2] §

[1]Departement Genie des Systemes d'Information
et de Telecommunication
Universite de Technologie de Troyes
Troyes Cedex, 10010, FRANCE
e-mail: Vidhyacharan.Bhaskar@utt.fr

[2]Department of Electrical and Computer Engineering
University of Alabama in Huntsville
Huntsville, AL 35899, USA
e-mail: ljoiner@ece.uah.edu

Abstract: Scheduled dataflow architecture (SDF) executes instructions in the prescribed order in which they arrive, even though data might already be available for executing the instructions. In this paper, we model the SDF using an open queuing network with feedback. Two models are proposed. The first model is comprised of a network of single queues each with a dedicated processor (server), while the second model is comprised of multiple servers for a single queue. A simulation is performed using mean value analysis on the network of queues. The average response time is computed for different numbers of synchronization and execution units in the processor. The paper also provides a comparison between the utilization of the queuing model with multiple servers and the queuing model with a single server. The two models depict the scheduling of instructions in the dataflow architecture in an efficient

Received: June 6, 2004

§Correspondence author

manner.

AMS Subject Classification: 68M20
Key Words: threads, traffic intensity, average queue lengths, average response times, average number of jobs, network utilization

1. Introduction

Scheduled dataflow architecture (SDF) shows promise as a new multithreaded data flow architecture to minimize the loss of CPU cycles due to memory latency [4, 3, 2]. SDF executes instructions in a prescribed order instead of executing them as soon as the data is available [4]. The expected order of instructions to be executed can be defined at compile time.

Primarily, there are two kinds of processors involved in SDF. They are the Synchronization Processor (SP) and the Execution Processor (EP). The SP is responsible for load/store operations. Data arrives at the SP in the form of threads (a set of instructions) from memory. The SP fetches the instructions from memory, schedules them, and sends them to the EP. The EP acquires the instructions from the SP, performs the arithmetic operations, and sends the result back to the SP, which then stores the results in memory. The SP is concerned with memory accesses and scheduling of threads, thus making them available to the EP [4]. SDF is a multithreaded architecture which requires greater thread level parallelism to achieve good performance, while superscalar architectures require greater instruction level parallelism to achieve good performance [2]. The performance of the SDF scales better with a proper balance of workload among the functional units (SPs and EPs) [2].

2. Open Queuing Network Model

An open queuing network approach is used to model the SDF. The open queuing network model can either have single server queues or multiple server queues. We will consider both models of the SDF in this paper.

2.1. Open Queuing Model with Feedback – Single Server

Consider the queuing model shown in Figure 1. The model contains m SPs and n EPs. Each SP and EP is analogous to a server with a single queue. The arrivals to the processors are in the form of threads. Each queue in the network

of m queues is served by an SP_i and is denoted as Q_{SP_i} $(1 \leq i \leq m)$, and each queue in the network of n queues served by an EP_j is denoted as Q_{EP_j} $(1 \leq j \leq n)$.

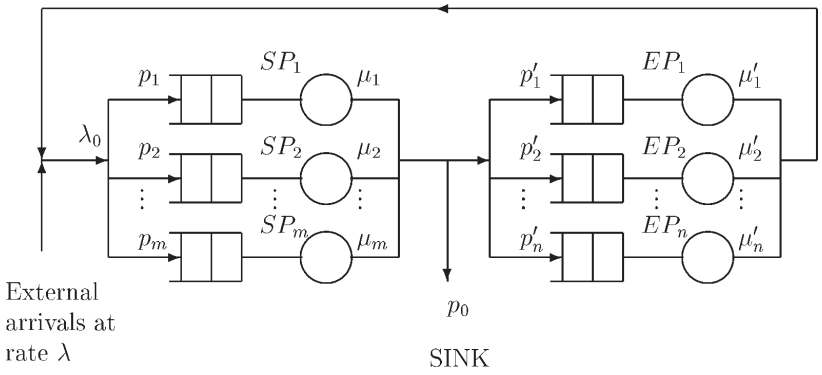

Figure 1: Open queuing model with feedback – single server

The model shown in Figure 1 is an open network of $(m + n)$ queues. Each queue is M/M/1 in nature. The first M in the M/M/1 notation denotes that the arrivals to the queue are memoryless. The arrivals are random in nature and they form a Poisson process at a constant arrival rate. The second M denotes that the service times are memoryless. The service times have an exponential distribution. The queuing discipline is first come first served (FCFS) [6]. There are no limits on the size of the queue or on the population from which the jobs come.

Arrivals to the SP nodes occur from outside the network at a rate λ, or through feedback. Let μ_1, μ_2, ..., μ_m be the service rates of the SPs, and let μ'_1, μ'_2, ..., μ'_n be the service rates of each of the EPs. Similarly, let λ_1, λ_2, ..., λ_m be the arrival rates to each of the SPs, and let λ'_1, λ'_2, ..., λ'_n be arrival rates to each of the EPs. Let the total arrival rate to all the SPs be λ_0.

From Figure 1,

$$\lambda'_j = \lambda_0 p'_j, \tag{1}$$

where p'_j is the probability of requesting service from EP_j $(1 \leq j \leq n)$.

The total arrival rate to all of the SPs is the sum of the external arrival rate and that coming from feedback,

$$\lambda_0 = \lambda + \sum_{j=1}^{n} \lambda'_j. \tag{2}$$

Rearranging (2), we have

$$\lambda_0 = \frac{\lambda}{1 - \sum_{j=1}^{n} p'_j}. \tag{3}$$

From Figure 1, we also have

$$p_0 + \sum_{j=1}^{n} p'_j = 1, \tag{4}$$

where p_0 is the probability that no service is requested from the EPs. Substituting (4) into (3), we have

$$\lambda_0 = \frac{\lambda}{p_0}. \tag{5}$$

The arrival rates to each of the SPs are

$$\lambda_i = \lambda_0 p_i = \frac{\lambda p_i}{1 - \sum_{j=1}^{n} p'_j} = \frac{\lambda p_i}{p_0}, \tag{6}$$

where p_i $(1 \leq i \leq m)$, is the probability of arrival at the i-th queue, requesting service from SP_i. The utilization of each SP_i is

$$\rho_i = \frac{\lambda_i}{\mu_i} = \frac{\lambda p_i}{\mu_i p_0}, \tag{7}$$

and the utilization of each EP_j is

$$\rho'_j = \frac{\lambda'_j}{\mu'_j} = \frac{\lambda p'_j}{\mu'_j p_o}, \tag{8}$$

where $1 \leq j \leq n$.

2.2. Performance Measures

The performance of the single server system is measured by the average queue lengths, average waiting times, average response times, and the average number of jobs in the system. Since all the queues in the model are assumed to be M/M/1, the average queue length in the SP is

$$E[N_i] = \frac{\rho_i}{1 - \rho_i} = \frac{\lambda p_i}{\mu_i p_0 - \lambda p_i}, \tag{9}$$

where $1 \leq i \leq m$. Similarly, the average queue length in the EP is

$$E\left[N'_j\right] = \frac{\rho'_j}{1 - \rho'_j} = \frac{\lambda p'_j}{\mu'_j p_0 - \lambda p'_j}, \tag{10}$$

where $1 \leq j \leq n$.

The average response time can be computed from Little's Theorem [6]. The average response times in the SPs and EPs are

$$E[R_i] = \frac{E[N_i]}{\lambda_0 p_i} = \frac{p_0}{\mu_i p_0 - \lambda p_i}, \tag{11}$$

where $1 \leq i \leq m$, and

$$E\left[R'_j\right] = \frac{E\left[N'_j\right]}{\lambda_0 p'_j} = \frac{p_0}{\mu'_j p_0 - \lambda p'_j}, \tag{12}$$

where $1 \leq j \leq n$. The relation $p_0 = \frac{\lambda}{\lambda_0}$ from (5) is used in (11) and (12).

The average waiting time at each queue, Q_{SP_i} $(1 \leq i \leq m)$ is

$$E[W_i] = E[R_i] - \frac{1}{\mu_i} = \frac{\lambda p_i}{\mu_i(\mu_i p_0 - \lambda p_i)}. \tag{13}$$

The average waiting time at each queue, Q_{EP_j} $(1 \leq j \leq m)$ is

$$E\left[W'_j\right] = E\left[R'_j\right] - \frac{1}{\mu'_j} = \frac{\lambda p'_j}{\mu'_j(\mu'_j p_0 - \lambda p'_j)}. \tag{14}$$

The average number of jobs in the system is the sum of the average queue lengths in the SPs and EPs,

$$E[N] = \sum_{i=1}^{m} E[N_i] + \sum_{j=1}^{n} E\left[N'_j\right] = \sum_{i=1}^{m} \frac{\lambda p_i}{\mu_i p_0 - \lambda p_i} + \sum_{j=1}^{n} \frac{\lambda p'_j}{\mu'_j p_0 - \lambda p'_j}. \tag{15}$$

2.3. Open Queuing Model with Feedback – Multiple Servers

The model shown in Figure 2 is an open queuing network of two queues. The SP is modeled as a queuing network of m servers, denoted as SP_i, $1 \leq i \leq m$, serving one queue, Q_{SP}. SP is a M/M/m queuing system with Poisson arrivals (or exponential inter-arrival time distribution), exponential service distribution, and m servers. Similarly, EP is a network of n servers, EP_j, $1 \leq j \leq n$, with one queue, Q_{EP}, and thus is a M/M/n queuing system. Again, there are no limits on the queue length or on the population from which the jobs come.

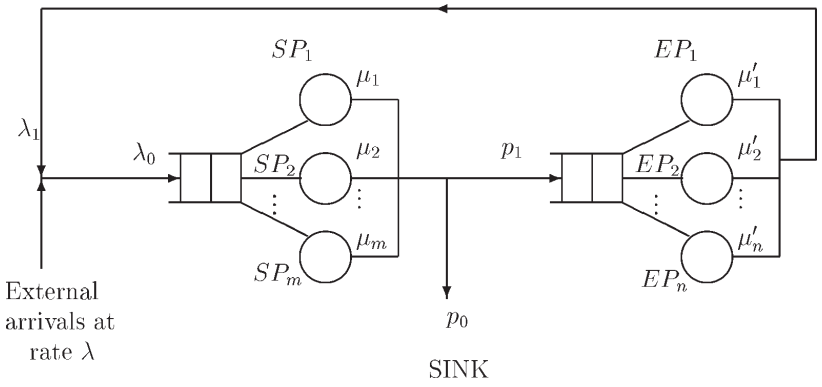

Figure 2: Open queuing model with feedback – multiple servers

Arrivals to the SPs occur either from outside at a rate λ, or through feedback. Let μ_1, μ_2, ..., μ_m be the service rates of each of the SPs, and let μ'_1, μ'_2, ..., μ'_n be the service rates of each of the EPs. Let λ_0 be the total arrival rate to Q_{SP}. Let λ_1 be the arrivals coming through feedback. For steady state to exist in this network, the total inflow must be equal to the total outflow,

$$\lambda_0 = \lambda + \lambda_1. \tag{16}$$

From Figure 2, we have

$$\lambda_1 = \lambda_0 p_1, \tag{17}$$

where p_1 is the probability of requesting service at any of the servers EP_j $(1 \leq j \leq n)$. Substituting (17) into (16), we have

$$\lambda_0 = \frac{\lambda}{1 - p_1}. \tag{18}$$

From Figure 2, we also have

$$p_0 + p_1 = 1. \tag{19}$$

Substituting (19) into (18), we have

$$\lambda_0 = \frac{\lambda}{p_0}. \tag{20}$$

Equation (20) must hold for steady state to exist. The utilization of each SP_i is

$$\rho_i = \frac{\lambda_0}{\mu_i} = \frac{\lambda}{p_0 \mu_i}, \tag{21}$$

where $1 \leq i \leq m$. The utilization of each EP_j is

$$\rho'_j = \frac{\lambda_1}{\mu'_j} = \frac{\lambda p_1}{p_0 \mu'_j}, \tag{22}$$

where $1 \leq j \leq n$.

2.4. Performance Measures

The M/M/m/∞ queuing system at time t is a birth-death process with the following parameters:

$$a_k = \lambda; \quad k \geq 0\,, \tag{23}$$

$$d_k = \begin{cases} k\mu, & 0 < k < m, \\ m\mu, & k \geq m. \end{cases} \tag{24}$$

In (24), a_k is the arrival rate to the M/M/m/∞ queuing system, and d_k is the departure rate from the M/M/m/∞ queuing system. Since there are infinite number of waiting positions in this queue, we consider an infinite number of jobs arriving at this queue. The departure rate depends on the number of busy servers at a time. If there are $k < m$ servers busy in the system, the average service (departure) rate is $d_k = k\mu$, and if all the m servers are busy in the system, $d_k = m\mu$. The departure parameter is state dependent. Note that the steady state probability of having no jobs in the system is

$$\pi_0 = \left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!\,(1-\rho)} \right]^{-1}. \tag{25}$$

The steady state probability of having k jobs in the system is

$$\pi_k = \begin{cases} \frac{(m\rho)^k}{k!}\pi_0, & k < m, \\ \\ \frac{\rho^k m^m}{m!}\pi_0, & k \geq m, \end{cases} \tag{26}$$

where $\rho = \frac{\lambda}{m\mu} < 1$. The ratio $\rho = \frac{\lambda}{m\mu}$ is the traffic intensity of the M/M/m/∞ queuing system (for SPs) and $\frac{1}{\mu}$ is the average service time of the SPs. A very similar result can be derived for the M/M/n/∞ queuing system (for EPs) by replacing m by n in (25) and (26), and by replacing ρ by ρ' (where $\rho' = \frac{\lambda}{n\mu'}$). ρ' is defined as the traffic intensity of the M/M/n/∞ queuing system and $\frac{1}{\mu'}$ is the average service time of the EPs [1].

The average queue length of the M/M/m/∞ queue is

$$L_m = \sum_{k=0}^{\infty} k\pi_k = m\rho + \frac{\rho\,(m\rho)^m}{m!\,(1-\rho)^2}\pi_0. \tag{27}$$

Similarly, the average queue length of the M/M/n/∞ queue is

$$L_n = n\rho' + \frac{\rho'\,(n\rho')^n}{n!\,(1-\rho')^2}\pi_0', \tag{28}$$

where

$$\pi_0' = \left[\sum_{k=0}^{n-1} \frac{(n\rho')^k}{k!} + \frac{(n\rho')^n}{n!\,(1-\rho')}\right]^{-1}. \tag{29}$$

The number of jobs in the system, L, is the sum of the average queue lengths of the individual queues,

$$L = L_m + L_n = m\rho + \frac{\rho\,(m\rho)^m}{m!\,(1-\rho)^2}\pi_0 + n\rho' + \frac{\rho'\,(n\rho')^n}{n!\,(1-\rho')^2}\pi_0'. \tag{30}$$

The average waiting time at Q_{SP} is

$$E\,[W_{SP}] = R_m - \frac{1}{\mu} = \frac{1}{\lambda}\left(m\rho + \frac{\rho\,(m\rho)^m}{m!\,(1-\rho)^2}\pi_0\right) - \frac{1}{\mu}. \tag{31}$$

The average waiting time at Q_{EP} is

$$E\,[W_{EP}] = R_n - \frac{1}{\mu'} = \frac{1}{\lambda}\left(n\rho' + \frac{\rho'\,(n\rho')^n}{n!\,(1-\rho')^2}\pi_0'\right) - \frac{1}{\mu'}. \tag{32}$$

Applying Little's Theorem, the average response time for the SPs is

$$R_m = \frac{L_m}{\lambda_0} = \frac{p_0}{\lambda}\left(m\rho + \frac{\rho\,(m\rho)^m}{m!\,(1-\rho)^2}\pi_0\right). \tag{33}$$

Similarly, the average response time for the EPs is

$$R_n = \frac{L_n}{\lambda_1} = \frac{p_0}{p_1\lambda}\left(n\rho' + \frac{\rho'\,(n\rho')^n}{n!\,(1-\rho')^2}\pi_0'\right). \tag{34}$$

Again applying Little's Theorem to the average number of jobs in the system, we have the response time of the system as

$$R = \frac{L}{\lambda} = \frac{1}{\lambda}\left(m\rho + \frac{\rho\,(m\rho)^m}{m!\,(1-\rho)^2}\pi_0 + n\rho' + \frac{\rho'\,(n\rho')^n}{n!\,(1-\rho')^2}\pi_0'\right). \tag{35}$$

2.5. Steady-State Probabilities

In this section, the expressions for the steady-state probability of having a certain number of jobs in the system for each of the models is presented.

2.5.1. Single-Server Case

The steady-state probability of having k_i jobs at node i in SPs (in an M/M/1 queue) is

$$\pi_i(k_i) = (1-\rho_i)\,\rho_i^{k_i}, \tag{36}$$

where $1 \le i \le m$. Similarly, the steady-state probability of having k'_j jobs at node j in EPs is

$$\pi_j\left(k'_j\right) = \left(1-\rho'_j\right)\left(\rho'_j\right)^{k'_j}. \tag{37}$$

From Jackson's Theorem [6], the joint probability of having k_i jobs at node i in SPs and k'_j jobs at node j (in an M/M/1 queue) is

$$\begin{aligned}\pi\left(k_1,k_2,\ldots,k_m,k'_1,k'_2,\ldots,k'_n\right) &= \prod_{i=1}^{m}\pi_i(k_i)\prod_{j=1}^{n}\pi_j\left(k'_j\right)\\ &= \prod_{i=1}^{m}(1-\rho_i)\,\rho_i^{k_i}\prod_{j=1}^{n}\left(1-\rho'_j\right)\left(\rho'_j\right)^{k'_j}.\end{aligned} \tag{38}$$

2.5.2. Multiple-Server Case

The joint probability distribution of having k jobs in SPs and k' jobs in EPs is

$$\pi\left(k,k'\right) = \pi_k\pi_{k'}, \tag{39}$$

where the π function is shown in (26).

3. Comparison of the Utilization of the Two Networks

Comparing (7) and (21), we have

$$\frac{(\rho_i)_s}{(\rho_i)_m} = p_i, \tag{40}$$

assuming that λ, μ_i, and p_0 are identical in both the networks. Here: $(\rho_i)_s$ refers to the utilization of SP node i in the single-server case, and $(\rho_i)_m$ refers to the utilization of SP node i in the multi-server case.

Similarly, comparing (8) with (22), we have

$$\frac{\left(\rho'_j\right)_s}{\left(\rho'_j\right)_m} = \frac{p'_j}{p_1}, \tag{41}$$

assuming that λ, μ'_j, and p_0 are identical in both the networks. Here: $\left(\rho'_j\right)_s$ refers to the utilization of EP node j in the single-server case, and $\left(\rho'_j\right)_m$ refers to the utilization of EP node j in the multi-server case.

The right-hand sides of (40) and (41) are less than one. Therefore,

$$(\rho_i)_s < (\rho_i)_m\,, \tag{42}$$

and

$$\left(\rho'_j\right)_s < \left(\rho'_j\right)_m\,. \tag{43}$$

Equation (40) shows that the efficiency of the servers in a multiple-server queuing system (with regard to SPs) is greater than that of the single-server queuing system (with regard to SPs) by a factor $\frac{1}{p_i}$, $(1 \leq i \leq m)$. This factor is a measure of how fast a server in a multiple-server queuing system can process jobs arriving at its queue as compared to a server in a single-server queuing system.

Equation (41) shows that the efficiency of the servers in a multiple-server queuing system (with regard to EPs) is greater than that of the single-server queuing system (with regard to EPs) by a factor $\frac{p_1}{p'_j}$, $(1 \leq j \leq n)$. This factor is a measure of how fast a server in a multiple-server queuing system can process jobs arriving at its queue as compared to a server in a single-server queuing system.

4. Simulation Results

The open network of queues was simulated using mean value analysis. The algorithm used is as given in [5]. The average response time is computed for the system of queues using M SPs and N EPs. The value of M is varied from 1 to 6 and the value of N is varied from 1 to 5. The average response times for different combinations of M and N was computed and is shown in Table 1.

No.	M	N	RT_s	RT_m
1	1	1	16.43	9.27
2	2	1	13.57	6.41
3	2	2	11.18	4.86
4	2	3	8.84	3.84
5	3	3	7.29	2.82
6	4	3	6.03	1.79
7	4	4	5.01	1.54
8	4	5	4.35	1.29
9	5	5	3.51	1.04
10	6	5	3.26	0.79

Table 1: Response time for single-server and multiple-server cases

Figure 3 is a plot of the average response time versus the total number of processors. The average response time is found to decrease gradually as the number of processors increases. The response times depend on the service rates of each of the processors used.

5. Conclusions

In summary, we have proposed two methods of modeling scheduled dataflow architecture. One model incorporates a single-server for each of its queues, and the other incorporates multiple servers in each of its queues. It is found that if the arrival rate, service rate, and the probability of leaving the network in both cases remain the same, then the multiple-server model gives a higher utilization in each of its nodes compared to the single-server model. Thus, the multiple-server model performs better than the single-server model. We have also performed mean value analysis on the single and multiple server cases, and found the response times for the multiple server network to be less than that of the single server network. When the degree of parallelism is high, SDF outperforms superscalar systems which do not scale well with increasing number of functional units.

References

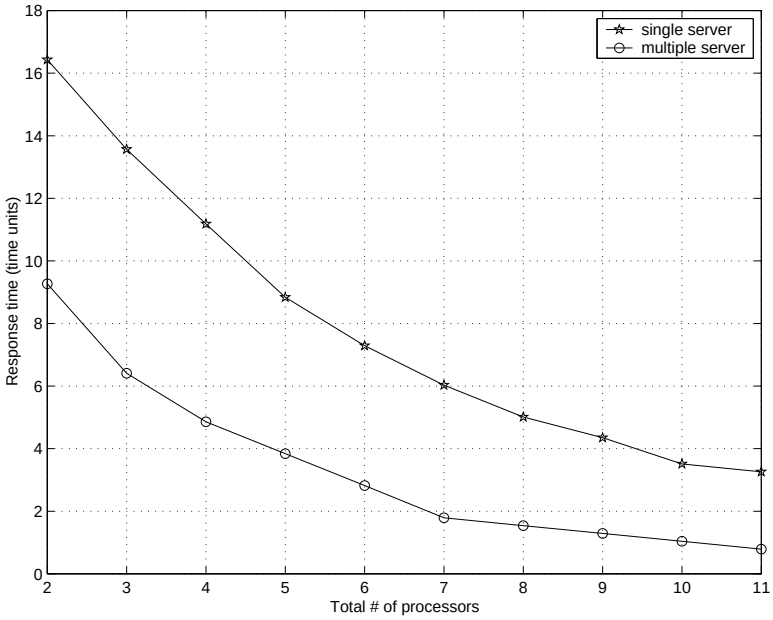

Figure 3: Response time vs No. of processors

[1] D. Gross, C.M. Harris, *Fundamentals of Queuing Theory*, Wiley series in Probability and Mathematical Statistics (1974).

[2] K.M. Kavi, J. Arul, R. Giorgi, Execution and cache performance of the scheduled dataflow architecture, *Journal of Universal Computer Science* (October, 2000).

[3] K.M. Kavi, R. Girogi, J. Arul, Scheduled dataflow: Execution paradigm, architecture, and performance evaluation, *IEEE Transactions on Computers*, **50** (2001), 834-846.

[4] K.M. Kavi, H. Kim, A.R. Hurson, Scheduled dataflow architecture: A synchronous execution paradigm for dataflow, *IASTED Journal of Computers and Applications*, **21** (1999), 114-124.

[5] M.K. Molloy, *Fundamentals of Performance Modeling*, New York, Macmillan Publishing Company (1989).

[6] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, New Jersey, Prentice-Hall (1982).